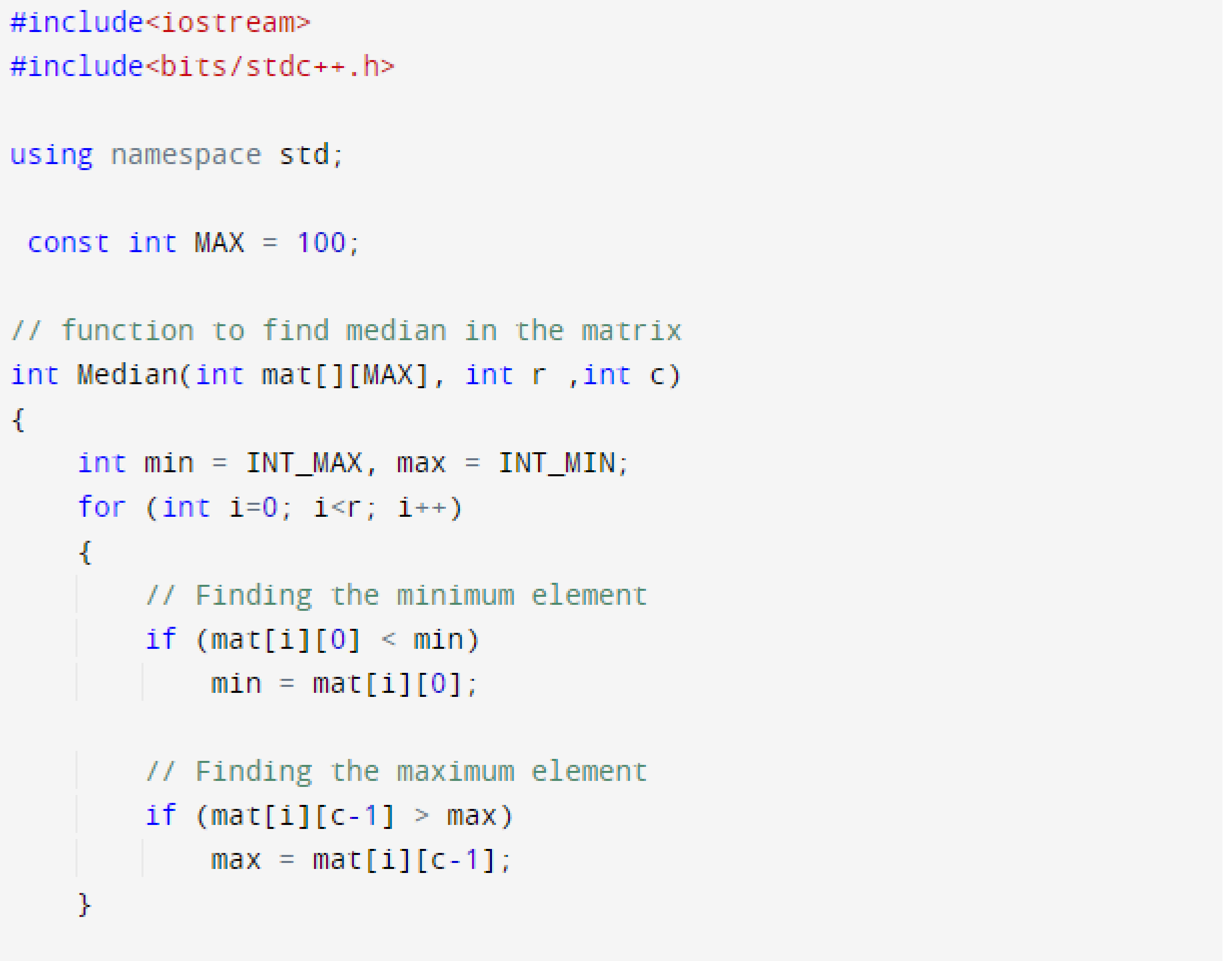
DAA Assignment -1

(Implemented the following problems using C++)

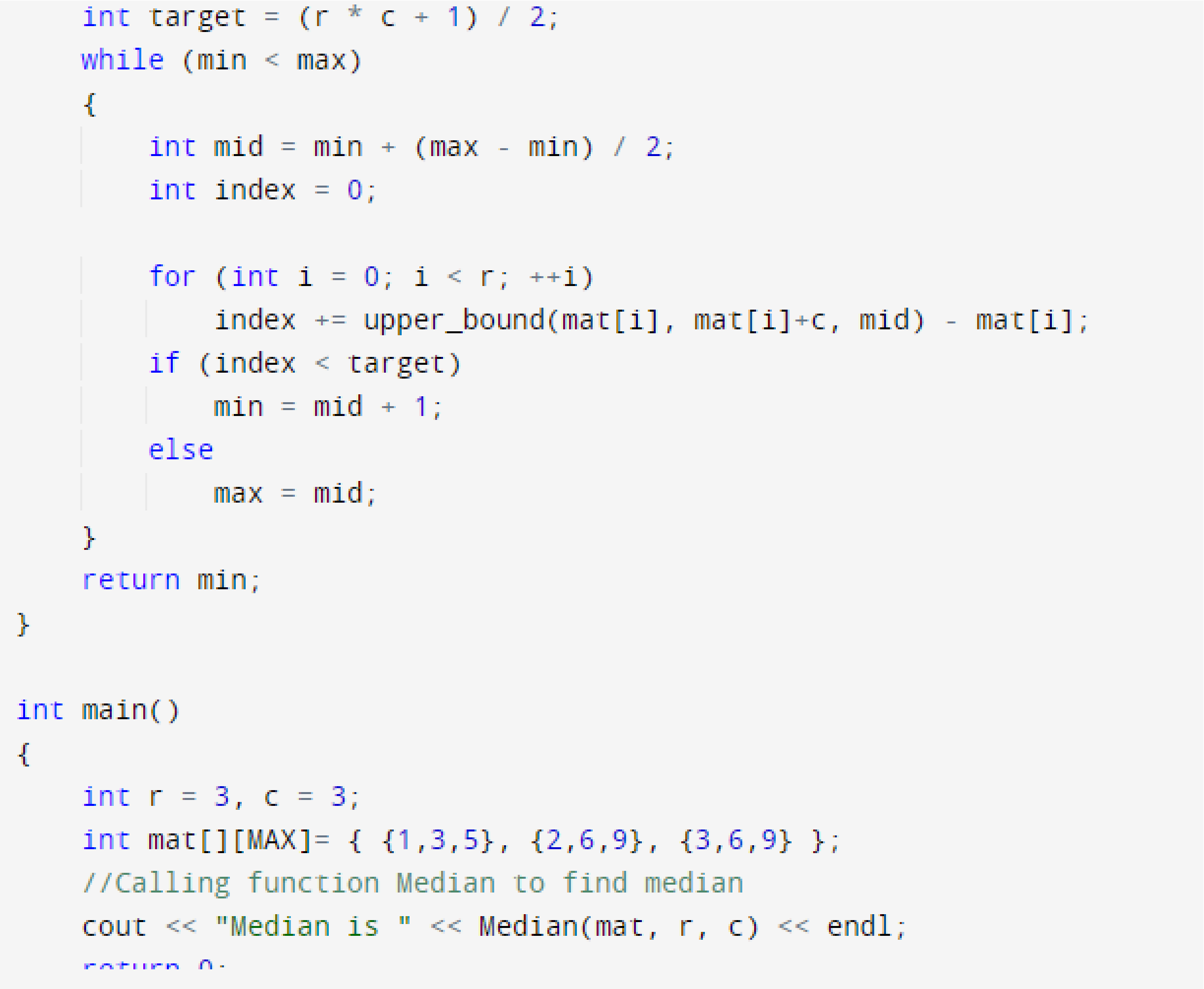
1 .Given a row wise sorted matrix of size **R*C** where R and C are always **odd**, find the median of the matrix. **5Mark**

- **Constraints:**
  1 <= R, C <= 400
  1 <= matrix[i][j] <= 2000

***DRIVER CODE OF THE PROBLEM STATEMENT*:**



```
#include<iostream>
#include<bits/stdc++.h>

using namespace std;

 const int MAX = 100;

// function to find median in the matrix
int Median(int mat[][MAX], int r ,int c)
{
    int min = INT_MAX, max = INT_MIN;
    for (int i=0; i<r; i++)
    {
        // Finding the minimum element
        if (mat[i][0] < min)
            min = mat[i][0];

        // Finding the maximum element
        if (mat[i][c-1] > max)
            max = mat[i][c-1];
    }
```

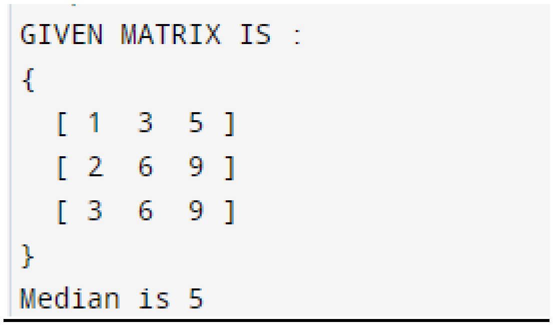


```cpp
    int target = (r * c + 1) / 2;
    while (min < max)
    {
        int mid = min + (max - min) / 2;
        int index = 0;

        for (int i = 0; i < r; ++i)
            index += upper_bound(mat[i], mat[i]+c, mid) - mat[i];
        if (index < target)
            min = mid + 1;
        else
            max = mid;
    }
    return min;
}

int main()
{
    int r = 3, c = 3;
    int mat[][MAX]= { {1,3,5}, {2,6,9}, {3,6,9} };
    //Calling function Median to find median
    cout << "Median is " << Median(mat, r, c) << endl;
    return 0;
```

Test Case 1:

```
Input:
R = 3, C = 3
M = [[1, 3, 5],
     [2, 6, 9],
     [3, 6, 9]]
Output: 5
Explanation: Sorting matrix elements gives
us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.
```

STD - OutPut:



```
GIVEN MATRIX IS :
{
  [ 1  3  5 ]
  [ 2  6  9 ]
  [ 3  6  9 ]
}
Median is 5
```

Test Case 2:

Input:

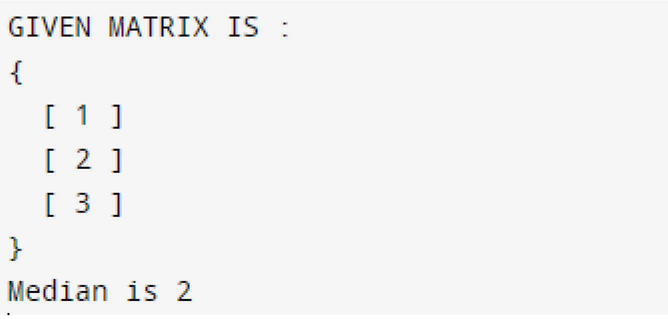
R = 3, C = 1

M = [[1], [2], [3]]

Output: 2

Explanation: Sorting matrix elements gives us {1,2,3}. Hence, 2 is median.

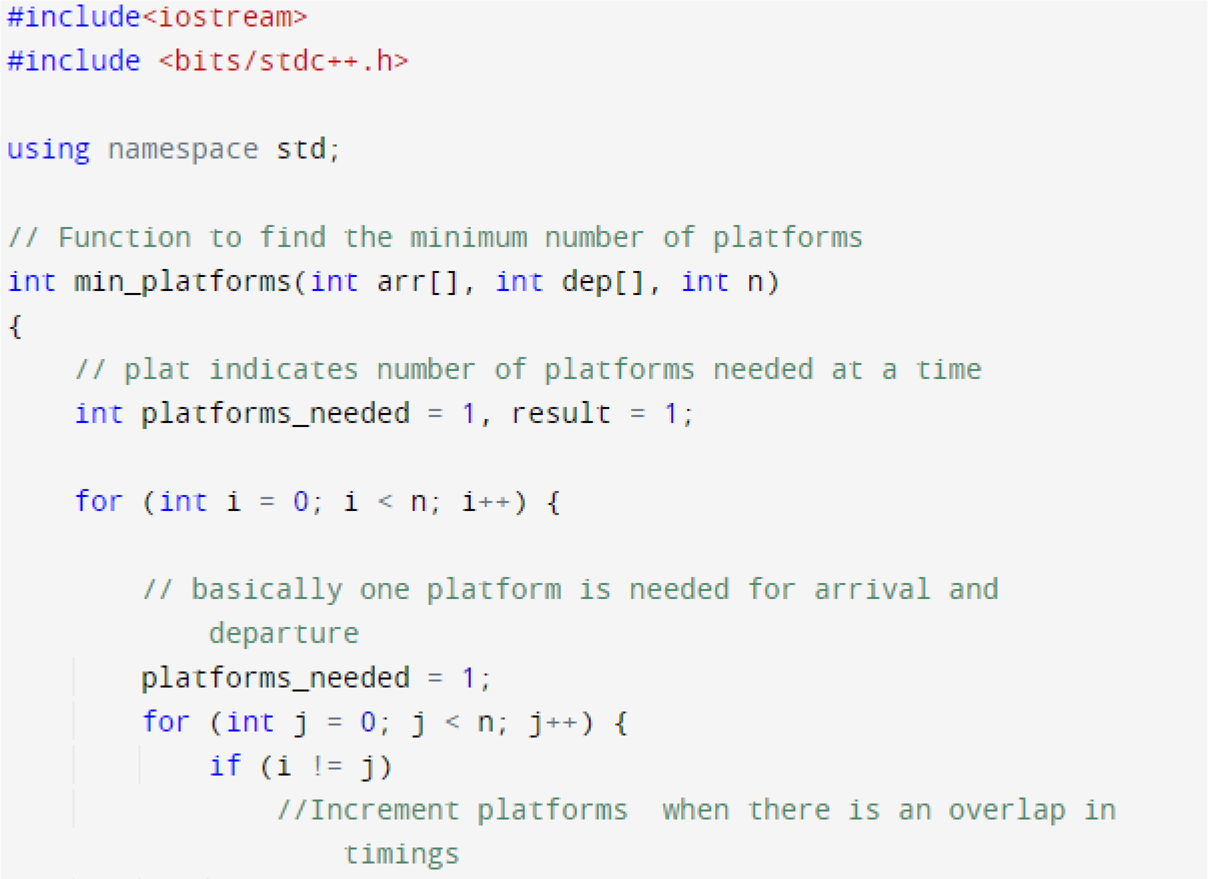
STD - OutPut :



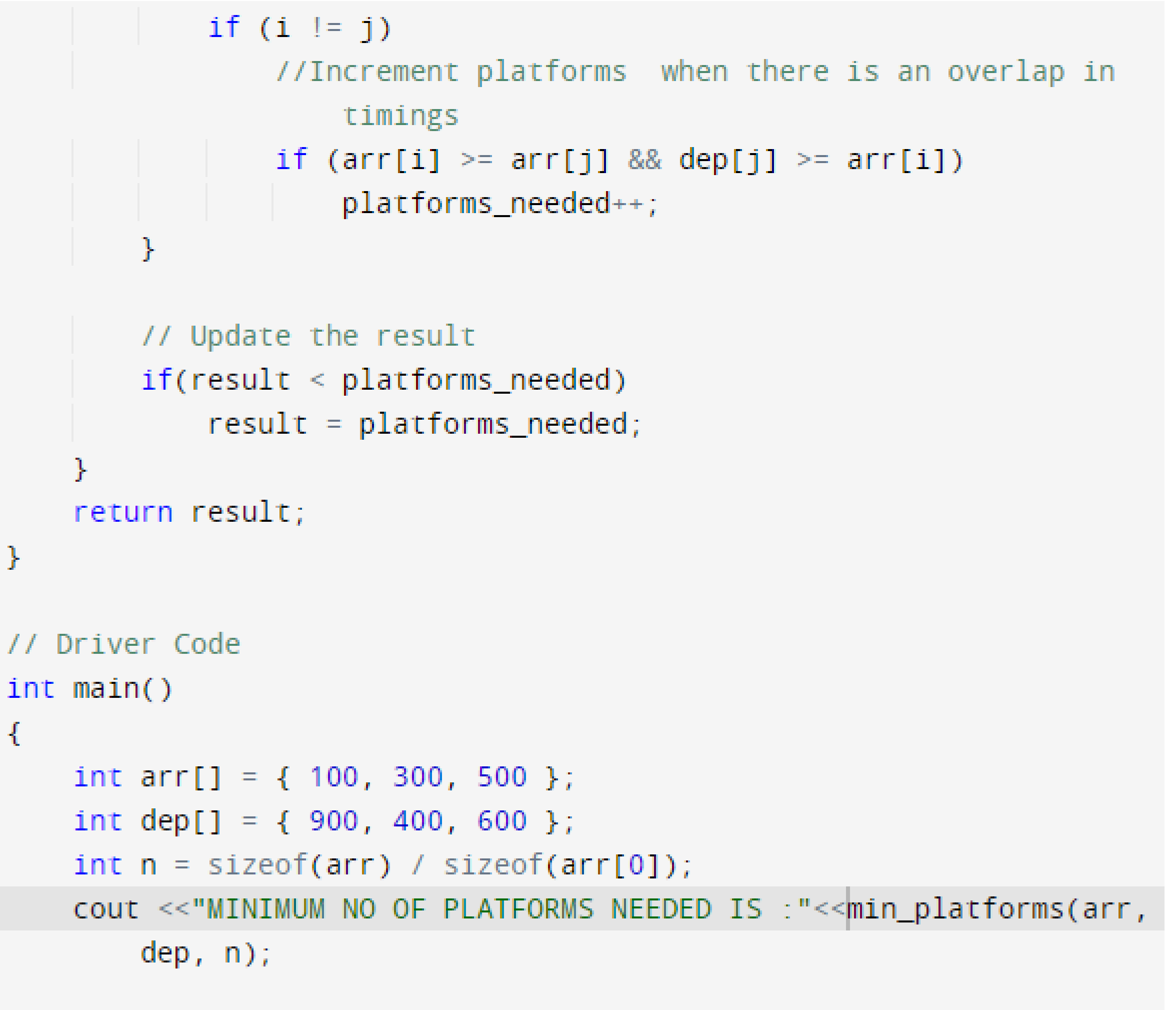
```
GIVEN MATRIX IS :
{
  [ 1 ]
  [ 2 ]
  [ 3 ]
}
Median is 2
```

2. Given the arrival and departure times of all trains that reach a railway station, the task is to find the minimum number of platforms required for the railway station so that no train waits. We are given two arrays that represent the arrival and departure times of trains that stop. **5Marks**

***DRIVER CODE FOR THE PROBLEM STATEMENT*:**



```
#include<iostream>
#include <bits/stdc++.h>

using namespace std;

// Function to find the minimum number of platforms
int min_platforms(int arr[], int dep[], int n)
{
    // plat indicates number of platforms needed at a time
    int platforms_needed = 1, result = 1;

    for (int i = 0; i < n; i++) {

        // basically one platform is needed for arrival and
            departure
        platforms_needed = 1;
        for (int j = 0; j < n; j++) {
            if (i != j)
                //Increment platforms  when there is an overlap in
                    timings
```



```
            if (i != j)
                //Increment platforms  when there is an overlap in
                    timings
                if (arr[i] >= arr[j] && dep[j] >= arr[i])
                    platforms_needed++;
        }

        // Update the result
        if(result < platforms_needed)
            result = platforms_needed;
    }
    return result;
}

// Driver Code
int main()
{
    int arr[] = { 100, 300, 500 };
    int dep[] = { 900, 400, 600 };
    int n = sizeof(arr) / sizeof(arr[0]);
    cout <<"MINIMUM NO OF PLATFORMS NEEDED IS :"<<min_platforms(arr,
        dep, n);
```

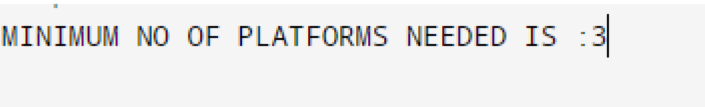
Test case 1 :

Input: *arr[] = {9:00, 9:40, 9:50, 11:00, 15:00, 18:00}, dep[] = {9:10, 12:00, 11:20, 11:30, 19:00, 20:00}*

Output: *3*

Explanation: *There are at-most three trains at a time (time between 9:40 to 12:00)*

STD OUTPUT:



```
MINIMUM NO OF PLATFORMS NEEDED IS :3
```

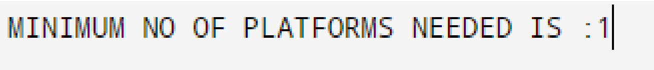
Test case 2 :

Input: *arr[] = {9:00, 9:40}, dep[] = {9:10, 12:00}*

Output: *1*

Explanation: *Only one platform is needed.*

***STD OUTPUT*:**



```
MINIMUM NO OF PLATFORMS NEEDED IS :1
```